Amendments to the Specification:

[0001] This application claims priority to ~~U.S. Utility patent application Ser. No. 09/386,264 filed Aug. 31, 1999, entitled "SYSTEM AND METHOD FOR TRANSMITTING A DIGITAL IMAGE OVER A COMMUNICATION NETWORK", and U.S. Provisional Patent Application No. 60/198,017, filed Apr. 18, 2000, entitled "LOSSLESS PROGRESSIVE STREAMING OF IMAGES OVER THE INTERNET", the entirety of which are both incorporated herein by reference.~~ U.S. Patent Application Serial No. 09/386,264, filed August 31, 1999, entitled "System and Method for Transmitting A Digital Image Over A Communication Network", now U.S. Patent No. 6,314,452, and U.S. Provisional Patent Application No. 60/198,017, filed April 18, 2000, entitled "Lossless Progressive Streaming of Images Over the Internet", both of which are incorporated herein by reference in their entirety.

[0004] In a narrow bandwidth environment, a simple transfer to the client computer of any original image stored in the server's storage is obviously time consuming. In many cases the user only wishes to view a low resolution version of the image and perhaps ~~a few more~~ several high-resolution details, in these instances it would be inefficient to transfer the full image. This problem can be overcome by storing images in ~~some~~ a compressed ~~formats~~ format. Examples [[for]] of such formats [[are]] include standards such as Progressive JPEG (W. Pennebaker and J. Mitchel, "JPEG, still image data compression standard", VNR, 1993) or the upcoming JPEG2000 (D. Taubman, "High performance scalable image compression with EBCOT", preprint, 1999). These formats allow progressive transmission of an image such that the quality of the image displayed at the client computer improves during the transmission.

[0005] In some ~~application~~ applications such as medical imaging, it is also necessary that whenever the user at the client computer is viewing a portion of the highest resolution of the image, the progressive streaming will terminate at lossless quality. This means that at the end of progressive transmission the pixels rendered on the screen are exactly the pixels of the original image. The current known "state-of-the-art" wavelet algorithms for progressive lossless streaming all have a major drawback: their rate-distortion behavior is [[very]] inferior to the "lossy" algorithms. The implications ~~are serious~~ of this include:

[0006] 1. Whenever the user is viewing any low resolution version of the image (at low resolutions the term "lossless" is not well defined) more data needs to be sent for the same visual quality.

[0008] Researchers working in this field are troubled by these phenomena. [[As]] F. Sheng, A. Bilgin, J. Sementilli and M. W. Marcellin [[say]] state in ~~[SBSM]~~ "Lossy to Lossless Image Compression Using Reversible Integer Wavelet Transform", Proc. IEEE International Conf. On Image Processing, 1998: ". . . Improved lossy performance when using integer transforms is a pursuit of our on-going work." ~~Here is an example:~~ An example is provided below in Table 1.

[0009] As ~~one~~ can ~~see~~ be seen from Table 1, state of the art progressive lossless coding is inferior to lossy coding by more than 1 dB at [[the]] high bit-~~rate~~ rates.

[0011] The main problem with known lossless wavelet algorithms, such as Set Partitioning in Hierarchical Trees (SPIHT) ~~[SP1]~~ A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning", IEEE Trans. Circuits and Systems for Video Tech. 6 (1996), 243-250 and compression with reversible embedded wavelets (CREW) ~~[ZASB]~~ A. Zandi, J.D. Allen, E.L. Schwartz and M. Boliek, "CREW: Compression with reversible embedded wavelets", Proc. of Data Compression Conference (Snowbird, Utah), 212-221, 1995, is that they use special "Integer To Integer" transforms (see "Wavelet transforms that map integers to integers", A. Calderbank, I. Daubechies, W. Sweldens, B. L. Yeo, J. Fourier Anal. Appl., 1998). These transforms mimic "mathematically proven" transforms that work well in lossy compression using floating-point arithmetic implementations. ~~But because~~ Because they are ~~constraint~~ constrained to be lossless, they do not approximate their related floating-point ~~ancestors~~ algorithms sufficiently well. Although in all previous work there have been attempts to correct this approximation in the progressive coding stage of the algorithm, the bad starting point, an inefficient transform, prevented previous authors ~~to obtain decent~~ from obtaining acceptable rate-distortion behavior.

~~[0012] Our algorithm solves the rate-distortion behavior problem. Using the fact that images are two-dimensional signals, we introduce new 2D lossless Wavelet transforms that approximate much better their lossy counterparts. As an immediate consequence our lossless progressive coding algorithm has the same rate-distortion of a lossy algorithm during the lossy part of the progressive transmission.~~

[0013] The system and method of the present invention solves the rate-distortion behavior problem. Using the fact that images are two-dimensional signals, novel 2D lossless Wavelet transforms are disclosed that better approximate their lossy counterparts. As an immediate consequence the lossless

progressive coding algorithm of the present invention has the same rate-distortion of a lossy algorithm during the lossy part of the progressive transmission.

[0013.1] The imaging system that is described below is directed to a lossless image streaming system that is different from traditional compression systems and overcomes the above problems. By utilizing a lossless means of progressive transmission, ~~means~~ the pixels rendered on the screen at the end of transmission are exactly the pixels of the original image that ~~was~~ were transmitted. The imaging system disclosed herein eliminates the ~~necessity~~ need to store a compressed version of the original image, by streaming ROI data using the original stored image. The imaging system of the present invention also avoids the computationally intensive task of compression of the full image. Instead, once a user wishes to interact with a remote image, the imaging server performs a fast preprocessing step in near real time after which it can respond to any ROI requests also in near real time. When a ROI request arrives at the server, a sophisticated progressive image-encoding algorithm is performed, but not for the full image. Instead, the encoding algorithm is performed only for the ROI. Since the size of the ROI is bounded by the size and resolution of the viewing device at the client and not by the size of the image, only a small portion of the full progressive coding computation is performed for a local area of the original image. This local property is also true for the client. The client computer performs decoding and rendering only for the ROI and not for the full image. This real time streaming ~~or Pixels-On-Demand™~~ architecture (known commercially as Pixels-On-Demand™) requires different approaches even to old ideas. For example, similarly to some prior art, the present imaging system is based on wavelets. But while in other systems wavelet bases are selected according to their coding abilities, the choice of wavelet bases in the present imaging system depends more on their ability to perform well in ~~the~~ a real time framework. The system of the present invention supports several modes of progressive transmission: by resolution, by accuracy and by spatial order.

[0041] The following notation is used throughout this document.

~~Term~~	~~Definition~~
~~4D~~	~~Four-dimensional~~
~~FIR~~	~~Finite Impulse Response~~
~~FWT~~	~~Forward Wavelet Transform~~
~~GUI~~	~~Graphical User Interface~~
~~ID~~	~~Identification tag~~
~~IWT~~	~~Inverse Wavelet Transform~~
~~ROI~~	~~Region Of Interest~~

~~URL~~	~~Uniform Resource Locator~~
~~LSB~~	~~Least Significant Bit~~
~~RMS~~	~~Root Square Error~~
~~FP~~	~~Floating Point~~
~~PDF~~	~~Probability Distribution Function~~

Term	Definition
1D	One dimensional
2D	Two dimensional
4D	Four dimensional
CDF	Cumulative Distribution Function
CD-ROM	Compact Disc-Read Only Memory
CREW	Compression with Reversible Embedded Wavelets
DVD	Digital Versatile Disc
EBCOT	Embedded block coding with optimal truncation
FIR	Finite Impulse Response
FP	Floating Point
FWT	Forward Wavelet Transform
GUI	Graphical User Interface
ID	Identification tag
IEEE	Institute of Electrical and Electronic Engineers
IWT	Inverse Wavelet Transform
JPEG	Joint Picture Entertainment Group
LSB	Least Significant Bit
PC	Personal Computer
PDF	Probability Density Function
RMS	Root Mean Square
ROI	Region Of Interest
SPHIT	Set Partitioning in Hierarchical Trees
URL	Uniform Resource Locator

[0047] With reference to FIG. 2, the system workflow is described. Using any browser type application, the user of the client computer 110 connects to the Web Server 140 or directly to the Imaging server 120 as described in FIG. 1. He/she then selects, using common browser tools, an image residing on the Image file storage 122. The corresponding URL request is received and processed by the Imaging Server 120. In case results of ~~previous computations~~ previous computations on the image are not present in the Imaging Cache 121, the server performs a fast preprocessing algorithm (see §2.1) in a lossless mode. The result of this computation is inserted into the cache 121. Unlike ~~other more "traditional"~~ prior art applications or methods which perform full progressive encoding of the image [[in]] using an "offline" type method, the goal of the

preprocessing step is to allow the server, after a relatively fast computational step, to serve any ROI specified by the user of the client computer. For example, for a ~~15M~~ 15 megabyte grayscale medical image, using the described (software) server, installed on a computer with a Pentium processor, fast disk, running Windows NT, the preprocessing step 501 will typically take 3 seconds. This is [[by]] an order of magnitude faster than [[a]] prior art "fill" compression ~~algorithm~~ algorithms such as ~~[S], [SP1], [T]~~ J. M. Shapiro, "An embedded hierarchical image coder using zero-trees of wavelet coefficients", IEEE Trans. Sig. Proc. 41 (1993), 3445-3462; A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning", IEEE Trans. Circuits and Systems for Video Tech. 6 (1996), 243-250; and D. Taubman, "High performance scalable image compression with EBCOT", IEEE Transactions on Image Processing 9 (2000), 1151-1170. Serving ~~ROI~~ ROIs is sometimes referred to as ~~called~~ "pixels-on-demand" which means a progressive transmission of any ROI of the image in "real-time", where the quality of the view improves with the transfer rate until a lossless view is received [[in]] on the client side. Once the preprocessing stage is done, the server sends ~~to the client~~ a notification message to the client that the "image is ready to be served". The server also transmits the basic parameters associated with the image such as dimensions, color space, etc. Upon receiving this notification, the client can select any ROI of the image using standard GUI. The ROI is formulated in step 203 into a request list that is sent to the server. Each such request corresponds to a data block ~~(§4)~~ as described in more detail in Section 4 hereinbelow. The order of requests in the list corresponds to some progressive mode selected in the context of the application such as "progressive by accuracy" rendering of the ROI. Upon receiving the ROI request list, the server processes the requests according to their order. For each such request the server checks if the corresponding data block exists in the ~~Cache~~ cache 121. If not, the server then computes the data block, stores it in the cache and immediately sends it to the client. Once a data block that was requested arrives at the client, it is inserted into the cache 111. At various points in time during the transfer process, a decision rule invokes a rendering of the ROI. Obviously, if some of the data blocks required for a high quality rendering of the ROI, were requested from the server, but have not arrived yet, the rendering of the ROI will be of lower quality. But, due to the progressive request order, the rendering quality will improve with each received data block in an "optimal" way. In case the user changes the ROI, the rendering task at the client is canceled and a new request list corresponding to a new ROI, sent from the client to the server, will notify the server to terminate the previous computation and transmission task and begin the new one.

[0049] Several benefits of ~~We will now explain in detail why~~ the rate-distortion behavior of ~~our~~ the progressive lossless algorithm ~~is better than other known algorithms~~ of the present invention are discussed below. Lossless wavelet ~~transform~~ transforms, must be integer-to-integer ~~transform~~ transforms, such that round-off errors are avoided. In order to demonstrate the difference between lossy and lossless transforms, let us look at the simplest wavelet, the Haar wavelet. Let x(k) be the k-th component of the one-dimensional discrete signal x. The first forward Haar transform step, in its accurate "mathematical" form, is defined by:

[0050] Where s is a low-resolution version of x, and d is the "difference" between s and x. In the case of lossless transform, applying the above transform results in round-off error. One possibility is to apply the transform step suggested by ~~[CDSI]~~ A. Calderbank, I. Daubechies, W. Sweldens and B. L. Yeo, "Wavelet transforms that map integers to integers", Applied and Computational Harmonic Analysis 5 (1998), 332-369:

[0051] The ~~symbol~~ notation $\lfloor \circ \rfloor$ [[is]] ~~the floor~~ denotes a round-off function that assigns the closest integer in the direction of the origin (i.e. zero) to a floating point number ~~meaning "greatest integer less than or equal to $\circ$"~~, e.g.

[0052] The one-dimensional transform step is generalized to a 2D separable transform step[[,]] by applying the 1D transform step twice, first in the X-direction and than (on the first stage output) in the Y-direction as described in FIG. 18, 19 and 20 ~~(see also [M] 7.7)~~. The full 2D Wavelet transform is applied by using the 2D Wavelet transform step iteratively in the classic Mallat decomposition of the image (FIG. 21) ~~([M] 7.7)~~. See S. Mallat, A wavelet tour of signal processing, Academic Press, 1998, Section 7.7.

[0054] 1. Reversibility, i.e., one can restore x(2n)and x(2n+1), by knowing s(n) and ~~d (n)~~ d(n), as follows:

[0056] then the least significant bit of s(n) and d(n) would have been the same and saved twice. In other words, there is a correlation between s(n) and d(n) in (3.4). From the view point of coding this should be avoided since there is a redundancy [[of]] in transmitting this bit.

[0057] On the other hand, the important scaling property, ~~that is a very important one,~~ is not kept in (3.2). Observe that the value of s(n) computed by (3.2), is smaller than its "real mathematical value"

as computed in (3.1), by factor of {square root}{square root over (2)}. Since s(n) should be rounded to an integer number, the fact that s(n) is smaller than what it should be, increases the round-off error. In low resolutions, the error is accumulated through the wavelet steps.

[0058] If we take the error as a model of "white noise" added to the i--th resolution in a multi-resolution representation of the image, i.e. X_i in FIG. 21, it can be proved that the variance of this noise exponentially increases as a function of i. This "contamination" to the multi-resolution image ~~damages~~ reduces the coding efficiency at low bit-rates. Let us describe this in detail for the case of the Haar wavelet. We have two assumptions in our analysis:

[0059] The parity (least significant bit) of an arbitrary coefficient c, in any of the wavelet steps is a uniformly distributed random variable, i.e.

$$\Pr(c \equiv 0 \mod 2) = \Pr(c \equiv 1 \mod 2) = \frac{1}{2}. \qquad [[(3.5)]]$$

[0060] 2. This parity is independent of other parity of other coefficients ~~coefficient's parity~~ (Identically independent ~~distributed~~ distribution).

[0065] As described in **Error! Reference source not found.**18, 19 and 20, according to the step defined in (3.2), we first apply the X-direction step

$$s(k,n) = \left\lfloor \frac{x(k,2n+1) + x(k,2n)}{2} \right\rfloor, \qquad (3.5)$$

[0066] ~~Where~~ where e is a random variable with a probability ~~distribution~~ density function ~~(P.D.F.)~~ (PDF) $p(\cdot)$ defined by

$$\begin{cases} p(-0.5) = \Pr(e = -0.5) = \frac{1}{2}, \\ p(0) = \Pr(e = 0) = \frac{1}{2}. \end{cases} \qquad (3.6)$$

[0067] Therefore,

$$\begin{cases} p(-0.5) = \Pr(e = -0.5) = \frac{1}{2}, \\ p(0) = \Pr(e = 0) = \frac{1}{2}. \end{cases} \qquad ~~(3.6)~~$$

$$\mathrm{E}(e) = -\frac{1}{4}, \mathrm{Var}(e) = \frac{1}{16}. \qquad (3.7)$$

[0086] And corresponds to the LL_i subband.

[0106] The approximation error evaluation results are summarized in the following table where the error is the difference between the normalized (in L_2-norm) coefficients according to ~~[CDSI]~~ Calderbank et al. (referenced supra) reversible transform and the "mathematical" transform (defined in (3.1)).

[0107] ~~Assuming a low-bit rate transmission where only the coefficients whose absolute value belongs to the range $[2^b,2^{b+1})$ are encoded, for every resolution i, where i is greater than b(less or more). It must be noted that the large error implies a significant loss of coding efficiency.~~ Thus, we may conclude that the cause for the degradation in the performance of the prior art progressive image encoding algorithm (see Table 1 supra), is the error propagation across resolutions of the prior art lossless wavelet transforms.

[0113] The full 2D Wavelet transform is applied by using the 2D Wavelet transform step iteratively in the classic Mallat decomposition of the image (FIG. 21) ~~([M] 7.7)~~. See S. Mallat, A wavelet tour of signal processing, Academic Press, 1998, Section 7.7. As ~~mentioned before~~ previously described, the Wavelet coefficients in ~~our proposed~~ the transform of the present invention are all scaled, i.e. normalized in L_2-norm as the Wavelet coefficients computed in the accurate "mathematical" transform.

[0116] Are a coefficient pair ~~results~~ resulting from a reversible de-correlated 1D-wavelet step

[0121] Let us call the bit~~, which~~ located on the ~~write-side~~ right side of the floating point the "Half Bit". Observe that the Half Bit of $d^{FP}(n)$ is the LSB of $d^{(1)}(n)$. Therefore, an equivalent way to do this in an integer computation without loosing the Half-Bit is to first calculate first the LSB of $d^{(1)}(n)$ by

[0123] By saving d(n) and HalfBit(n) we can restore ~~back~~ $d^{(1)}(n)$ at a later time.

[0125] The Half bit matrix is hidden in the HH-subband in the description of FIG. 18, 19 and 20. It is described explicitly in the specification of the transform ~~as much~~ and in the coding algorithm.

[0160] e' and e" are random variables with ~~P.D.F.~~ Probability Density Function (PDF) defined in (3.6).

[0169] e'" is a random variable with ~~P.D.F.~~ PDF defined in (3.6).

[0174] The results indicate that at ~~The meaning of this result is that in a~~ low ~~bit rate~~ bit rates, where only large coefficients are encoded, ~~this~~ the error is negligible.

[0196] The lossless algorithm ~~receive~~ receives as input the following parameters:

Table 1 - Lossless Encoding Algorithm Input Parameters

Variable	Meaning
coef	Matrix of subband coefficients, containing $3\times\left(\frac{tileLength}{2}\right)^2$ coefficients
HalfBit	Matrix of bits containing $\left(\frac{tileLength}{2}\right)^2$ bits.

[0220] ~~Explanations to~~ Regarding the least significant bit encoding algorithm the following is noted:

[0227] Obviously, this algorithm is a reversed step of the encoding algorithm of section 5.1, performed in the server 120. The client computer 110 during the progressive rendering operation performs the decoding algorithm. Similar to the encoding algorithm, the decoding algorithm is described for an image with one component (such as a grayscale image), but of course could also be used with an image with more than one component. The input parameters to the lossless algorithm are given below:

TABLE [[2]] 4.1 - Lossless decoding algorithm input parameters

Variable	Meaning
coef	Empty matrix of subband coefficients to be filled by the decoding algorithm.
HalfBit	Matrix of bits containing $\left(\frac{tileLength}{2}\right)^2$ bits.

[0233] 1. Assign the value zero to each coefficient Coef (x,y) ~~the value zero~~.

[0234] 2. Assign the value zero to each bit ~~belongs~~ belonging to the HalfBit matrix ~~the value zero~~.

[0237] 5. If the "first" data block $(t\_x, t\_y, t\_resolution, maxBitPlane(t\_resolution))$ is available at the client, read the first byte, which is the value of $maxBitPlane(tile)$.

[0238] ~~Is available at the client, read the first byte, which is the value of maxBitPlane (tile).~~

[0245] ~~Explanations to~~ Regarding the pseudo code the following is noted:

[0246] 1. The decoder's method morecoef( ) scans all the coefficients in the HH, HL and LH subband. But, since the LH-subband is skipped in the encoding algorithm, ~~we don't call to~~ decodeSymbol( ) is not called for its coefficients. Instead ~~of this~~, ~~we update~~ their least significant bit ~~as~~ is initialized to zero.

[0252] The basic parameters of a ROI are worldPolygon and scale which determine uniquely the ROI view. If the ROI is to be rendered onto a viewing device with limited resolution, then a worldPolygon containing a large portion of the image will be coupled by a small scale. In the case where the rendering is done by a printer, the ROI could be a strip of a proof resolution of the original image that has arrived from the server computer 120. This strip is rendered in parallel to the transmission, such that the printing process will terminate with the end of transmission. The other view parameters determine the way in which the view will be rendered. The parameters deviceDepth and viewQuality determine the quality of the rendering operation. In cases where the viewing device is [[of]] low resolution or the user sets the quality parameter to a lower quality, the transfer size can be reduced significantly.

[0256] ~~However, since~~ Since lossless compression is ~~mostly~~ frequently required in medical images transmission, where typically more than 8 bits images are used, ~~we amplify our discussion on~~ the curve (luminanceMap hereinabove), which defines the mapping from the original image gray scale range (typically 10,12,16 bits) to an 8-bit screen, is discussed in more detail. ~~Further more~~ Furthermore, in viewing medical images ~~viewing~~, regardless of the original image depth, mapping is required in order to control the brightness and contrast of the image.

[0259] The curve influences not only the mapping, i.e. the ~~drawing~~ rendering to the screen, but also the request from the server. To understand ~~that~~ this, let us ~~concentrate in~~ focus on the maximal gradient of the curve (FIG. 17). In a lossy mode, the request is created such that the image approximation [[in]] on the client side is close enough to the original image, i.e., the RMS (Root Mean Square Error) is visually negligible. When a curve (i.e. mapping function) is applied, the RMS can be increased or reduced. The maximal RMS increasing factor depends on the maximal gradient of the curve as follows:

[0266] If the RMS increasing factor is greater than 1, it means that the "new RMS" may be greater than we consider as visually negligible error. Thus, the request list should be ~~increase~~ increased (i.e. more bit-planes should be requested from the server) in order to improve the approximation accuracy. Conversely, if the RMS increasing factor is smaller than 1, the request listing ~~can~~ should be reduced. The exact specification of this is given in the following section.

[0268] In step 402 using the ROI view parameters, the client imaging module at the client computer 110 calculates the data ~~blocks~~ block request list ordered according to the particular progressiveMode selected. Given the parameters worldPolygon and Scale, it may be determined which subband tiles in the "frequency domain" participate in the reconstruction of the ROI in the "time domain". These tiles contain all the coefficients that are required for an "Inverse Subband/Wavelet Transform" (IWT) step that produces the ROI. First, the parameter dyadicResolution (ROI) is computed, which is the lowest possible dyadic resolution higher than the resolution of the ROI. Any subband tiles of a higher resolution than dyadicResolution (ROI) do not participate in the rendering operation. Their associated data blocks are therefore not requested, since they are visually insignificant for the rendering of the ROI. If scale.gtoreq.1, then the highest resolution subband tiles are required. If scale.ltoreq.2.sup.1-numberOfRe- solutions then only the lowest resolution tile is required. For any other value of scale we perform the mapping described below in Table 6.

Table 6

scale	*highestSubbandResolution*
$scale \le 2^{1-numberOfResolutions}$	1
$2^{1-numberOfResolutions} < scale \le 1$	$numberOfResolutions - \lfloor -\log_2(scale) \rfloor$
$scale > 1$	*numberOfResolutions*

[0269] Once it has been determined which subband tiles participate in the rendering of the ROI, it is necessary to find which of their data blocks are visually significant and in what order they should be requested. Using well known rate/distortion rules from the field of image coding (such as is described in S. Mallat and F. Falzon, "Understanding image transform codes", Proc. SPIE Aerospace Conf., 1997), ~~it is not too difficult to determine~~ an optimal order can be determined in which the data blocks should be ordered by the client imaging module (and thus delivered by the server 120). This optimal order is described in steps 301-310 of FIG. 3 for the "Progressive By Accuracy" mode. The underlying mathematical principal behind this approach is "Non-Linear Approximation".

[0270] First, the subband coefficients with largest absolute values are requested since they represent the most visually significant data such as strong edges in the image. ~~Notice~~ Note that high resolution coefficients with large absolute ~~value~~ values are requested before low resolution coefficients with smaller absolute ~~value~~ values. Within each given layer of precision (bit plane) the order of request is according to resolution; low resolution coefficients are requested first and the coefficients of highestSubbandResolution are requested last.

[0271] The main difficulty of this step is this: Assume a subband tile is required for the rendering of the ROI. This means that $t\_resolution \le dyadicResolution(ROI)$ and the tile is required in the IWT procedure that reconstructs the ROI. It must be understood which of the data blocks associated with the subband tile represent visually insignificant data and thus should not be requested. Sending all of the associated data blocks will not affect the quality of the progressive rendering. However, in many cases transmitting the "tail" of data blocks associated with high precision is unnecessary since it will be visually insignificant. In such a case, the user will see that the transmission of the ROI from the server 120 is still in progress, yet the progressive rendering of the ROI ~~seems to~~ no longer ~~to change~~ changes the displayed image.

[0272] Additionally, the influence of the luminance mapping on the accuracy level of the requested data block [[as]] is described below. Supposing for some t_x,t_y and t_resolution, the set

[0273] Is requested where T is the minimal bit plane required to the current view. Here, where the luminance mapping is taken [[in]] into account, the value of T might be increased or decreased.

[0279] Image depth of ~~12-bit~~ 12-bits

[0280] Screen depth of ~~8-bit~~ 8-bits

[0281] Linear luminance mapping, ~~I.e.~~ i.e.,

[0285] Thus, one bit plane is added to the original set.

[0288] Each such structure represents the ~~n_xxn_y~~ n_x by n_y data blocks

[0289] The encoding algorithm attempts to create the shortest possible list of structures, collecting the data blocks ~~to~~ of the largest possible rectangles can ~~do~~ achieve this. It is important to note that the algorithm ensures that the order of data blocks in the request list is not changed, since the server 120 will respond to the request stream by transmitting data blocks in the order in which they were requested. A good example of when this works well is when a user zooms in into a ROI at a high resolution that was never viewed before. In such a case the request list might be composed of hundreds of requested data blocks, but they will be collected to one (x, y) rectangle for each pair (t_resolution, t_bitPlane).

[0291] The client computer 110, upon receiving ~~from the server computer 120~~ an encoded stream containing data blocks from the server computer 120, decodes the stream and inserts the data blocks into their appropriate location in the distributed database using their ID as a key. The simple decoding algorithm performed here is a reversed step of the encoding scheme described infra. Since the client 110 is aware of the order of the data blocks in the encoded stream, only the size of each data block need be reported along with the actual data. In case the server 120 ~~informs of~~ indicates an empty data block, the receiving module marks the appropriate slot in the database as existing but empty.

[0295] During the transmission of ROI data from the server to the client, the client performs rendering operations of the ROI. To ensure that these rendering tasks do not interrupt the transfer,

the client runs two program threads: communications and rendering. The rendering thread runs in the background and ~~draws into~~ uses a pre-allocated "off-screen" buffer. Only then does the client use device and system dependant tools to output the visual information from the "off-screen" to the rendering device such as the screen or printer.

[0302] As ROI data is transmitted to the client 110, the rendering algorithm is performed at certain time intervals of a few seconds. At each point in time, only one rendering task is performed for any given displayed image. To ensure that progressive rendering does not become a bottleneck, two rates are measured: the data block transfer rate and the ROI rendering speed. If it is predicted that the transfer will be finished before a rendering task, a small delay is inserted, such that rendering will be performed after all the data arrives. Therefore, in a slow network scenario (as the Internet often is), for almost the entire progressive rendering tasks, no delay is inserted. With the arrival of every few kilobytes of data, containing the information of a few data blocks, a rendering task visualizes the ROI at the best possible quality. In such a case the user is aware that the bottleneck of the ROI rendering is the slow network and has the option to accept the current rendering as a good enough approximation of the image and not wait for all the data to arrive.

[0304] This data-structure is required to efficiently store subband coefficients, in memory, during the rendering algorithm. This is required since the coefficients are represented in either long integer precision (i.e. lossless coding mode) or floating-point precision (i.e. lossy coding mode) ~~precision~~ which typically require more memory than pixel representation (1 byte). In lossy mode, the coefficients at the client side 110 are represented using floating-point representation, even if they were computed at the server side 120 using an integer implementation. This ~~will minimize~~ minimizes round-off errors.

[0307] Beginning with the lowest resolution 1, the algorithm proceeds with [[a]] recursive multiresolution ~~march~~ processing from the top of the ROI to bottom (i.e. y direction). Referring to FIGS. 10 and 11, in step 1101, the multiresolution strips are filled with sub-tiles of coefficients 1050 decoded from the database or read from the memory cache. ~~From the coefficients we obtain multiresolution~~ Multiresolution pixels 1051 are obtained from the coefficients using an inverse subband transform step 1102 (shown in further detail in FIG. 10). Each time a tile of pixels at resolutions ~~j<dyadicResolution (ROI)~~ j<dyadicResolution(ROI) is reconstructed, it is written into the pixel strip at the resolution j. Each time a tile of pixels at the highest resolution ~~dyadicResolution~~

~~(ROI)~~ dyadicResolution(ROI) is reconstructed, it is ~~fed into~~ input to the inverse color transform and resizing steps 1103,1104.

[0309] The subband coefficients data structure, described previously in section 6.5.2, is filled on a tile basis. Each such subband tile is obtained by decoding the corresponding data blocks stored in the database or by reading them from the memory cache. The memory cache is used to store coefficients in a simple encoded format. The motivation is this: the decoding algorithm described previously in section 5.2 is computationally intensive and thus should be avoided whenever possible. To this end the rendering module uses a memory cache 111 where subband ~~coefficient~~ coefficients are stored in a very simple encoded format which ~~decodes~~ can be decoded very ~~fast~~ quickly. For each required subband tile, the following extraction procedure is performed, described in FIG. 25, beginning at step 2501. In step 2502, if no data blocks are available in the database for the subband tile, its coefficients are set to zero (step 2503). In step 2504, if the tile's memory cache storage is updated, namely it stores coefficients in the same precision as in the database, then the coefficients can be efficiently read from there (step 2505). In step 2506, the last possibility is that the database holds the subband tile in higher precision. Then, the tile is decoded down to the lowest available bit plane using the algorithm previously described in section 5.2 and the cached representation is replaced with a representation of this higher precision information.

[0312] Remark: Recall from Section 5.1.1 that the "half bits" are initialized ~~as zeros~~ to zero, therefore the inverse step is well defined even if their "real" value is not available in the client yet.

[0316] In case the resolution of the ROI is not an exact dyadic resolution, the image obtained by the previous step must be re-sized to this resolution. This can be accomplished using operating system imaging functionality. In most cases the operating system's implementation is sub-sampling which produces in many cases an aliasing effect which is not visually ~~not~~ pleasing. To provide higher visual quality, the imaging system of the present invention may use ~~the~~ a method of linear interpolation, ~~for example~~ such as described in J. Proakis and D. Manolakis, "Digital signal processing", Prentice Hall, 1996. The output of the linear interpolation step is written to the off-screen buffer. From there it is displayed on the screen using system device dependant methods.

[0318] When luminanceMap is active, mapping to 8-bit screen is performed using the mapping function described in Section 6.1.1.

[0323] In step 503, the server reads from cache or encodes data ~~block~~ blocks associated with low resolution portions of the ROI, using the cached result of the preprocessing stage 501.

[0326] The preprocessing step is now described with respect to FIG. 6. The goal of the preprocessing ~~algorithm's goal~~ algorithm is to provide the fastest response to the user's request to interact with the image. Once this fast computational step is performed, the server is able to provide efficient "pixel-on-demand" transmission of any client ROI requests that [[will]] follow. In most cases the first ROI is a view of the full image at the highest resolution that "fits" the viewing device. The preprocessing algorithm begins with a request for an uncompressed image that has not been processed before or has been previously processed but the ~~result of this previous computation has~~ results of which have been deleted from the cache. As explained hereinabove, this unique algorithm replaces the possibly simpler procedure of encoding the full image into some progressive format. This latter technique will provide a much slower response to the user's initial request then the technique described below. At the end of the algorithm a "ready to serve ROI of the image" message is sent to the client containing basic information on the image. While some of this information, image dimensions, original color space, resolution etc., is available to the user of the client computer, most of this information is "internal" and required by the client to formulate ROI request lists (§6.2) and progressively render (§6.5). Next we describe in detail the preprocessing algorithm.

[0331] a. Coding performance (in a rate/distortion sense): This is ~~obviously required from~~ a requirement of any ~~decent~~ subband/wavelet transform.

[0333] c. Fast transform implementation: ~~Can~~ Whether the associated fast transform can be implemented using lifting steps (as described, for example, by I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps", J. Fourier Anal. Appl., Vol. 4, No. 3, pp. 247-269, 1998), using only integer shifts and additions, etc. Some good examples are the Haar and CDF transforms (1,3), (2,2) [[***]] described in I. Daubechies, "Ten lectures on wavelets", Siam, 1992.

[0335] e. Lossless mode: ~~IflosslessMode~~ If losslessMode is true, we must choose the filters from a subclass of reversible transforms (see, for example, "Wavelet transforms that map integers to integers", A. Calderbank, I. Daubechies, W. Sweldens, B. L. Yeo, J. Fourier Anal. Appl., 1998).

[0336] f. Low system I/O: If the network 130 in FIG. 1 connecting between the Image residing on the storage 122 and the imaging server 120 is slow, the bottleneck of the preprocessing stage ~~(and~~

and possibly the whole imaging system ~~for that fact)~~ might [[be]] simply be the reading of the original image. In such a case a transform may be chosen with a lazy sub-sampling low pass filter that corresponds to efficient selective reading of the input image. Many interpolating subband transforms with increasing number of vanishing moments can be selected to suit this requirement. However, this choice should be avoided whenever possible, since it conflicts with considerations (a) and (b).

[0347] Most prior art wavelet coding algorithms ~~have not addressed~~ do not address the problem of memory complexity. Usually ~~the authors have assumed~~ these prior art algorithms assume there is sufficient memory such that the image can be transformed in memory from ~~the~~ a time domain representation to a wavelet frequency domain representation. ~~It seems the~~ The upcoming JPEG2000 standard will likely address this issue, as did its predecessor, the JPEG standard. The preprocessing algorithm also requires performing subband transforms on large images, although not always on the ~~fall~~ full image, and thus requires careful memory management. This means that the memory usage of the algorithm is not [[of]] on the order of magnitude of the original image, as described in J. M. Shapiro, "An embedded hierarchical image coder using zero-trees of wavelet coefficients", IEEE Trans. Sig. Proc., Vol. 41, No. 12, pp. 3445-3462, 1993.

[0353] Referring to FIG. 6, during the preprocessing stage, the resolutions are scanned simultaneously from start to end in the y direction. For each color component and resolution, the corresponding strip stores low-pass coefficients or pixels at that resolution. The core of the preprocessing algorithm are steps 604-607, where tiles of pixels of length tileLength+2.times.maxFilterSize are read from the memory strips and handled one at a time. In step 604 the tile is transformed into a tile of length tileLength containing two types of coefficient data: subband coefficients and pixels at a lower resolution. The subband coefficients are processed in steps 605-606 and [[are]] stored in the cache. The lower resolution pixels are inserted in step 607 to a lower resolution memory strip. Whenever such a new sub-tile of lower resolution pixels is inserted into a strip, the ~~algorithm's~~ memory management module of the algorithm performs the following check: If the part of the sub-tile exceeds the current virtual boundaries of the strip, the corresponding first lines of the strip are considered unnecessary and ~~anymore. Their~~ their memory is (virtually) re-allocated for the purpose of storing ~~the sub-tile's~~ new sub-tile data.

[0355] This step [[is]] uses the conversion formula described in FIG. 13~~. This step~~ and must be performed before step 602, because the lossless color conversion is non-linear. ~~7.1.4 Step 602: Lossless Wavelet Low Pass~~

[0355.1] 7.1.4 Step 602: Lossless Wavelet Low Pass

[0356] The motivation for the low pass step is explained in § 6.1.4 in the above-cited ~~Ser.~~ Serial No. 09/386,264, which disclosure is incorporated herein by reference. ~~In a lossless mode there are a few emphasis that are represented here~~ Several important aspects of lossless mode are emphasized below.

[0357] In step 602, the low pass filters of the transforms $numberOfResolutions - jumpSize < j \le numberOfResolutions$, are used to obtain a low resolution strip at the resolution numberOfResolutions-jumpSize (as can be seen in FIG. 26). Typically, it is required to low pass filter about $2^{jumpSize}$ lines of the original image 1010 to produce one line of the low resolution image. The low pass filter calculation is initiated by ~~a read of~~ reading tiles of pixels from the memory strips performed in step 604. Whenever there is an attempt to read missing low resolution lines, they are computed by low ~~passing~~ pass filtering the original image and ~~inserted~~ inserting the results into the memory strip. The insertion over-writes lines that are no longer required[[,]] such that the algorithm is memory constrained. In the case where a non-linear color transform took place in the previous step 601, the results of that transform are ~~low passed~~ low pass filtered.

[0358] In the lossless mode of operation, the jumpSize parameter defines the number of lossless wavelet low pass filtering steps that should be done. A single low pass filtering step is the same for Haar and CDF (1,3) and defined by the following two stages (taken from (3.20) and (3.22):

[0362] ForjumpSize=1 and jumpSize=2 (other sizes ~~practically~~ typically are not needed), the server performs these steps efficiently (almost like the lossy algorithm) by a single operation that simulates exactly jumpSize low pass steps defined in (7.1). As noticed from (7.1), the simplicity of the formula makes filters such as Haar and CDF (1,3) "optimal" ~~in the~~ with respect [[of]] to low pass efficiency.

[0364] In Step 603 we perform one step of an efficient local lossless wavelet transform (§3), on a tile of pixels at the resolution $1 \le j \le numberOfResolutions - jumpSize$. The type of transform is

determined by the parameter lossless WaveletTransformType(j). As described in FIG. 1, the transform is performed on an "extended" tile of pixels [[is]] of length $tileLength + 2 \times maxFilterSize$ (unless we are at the boundaries), read directly from a multi-resolution strip at the resolution j+1. The output of ~~the~~ this step is a lossless subband tile composed of wavelet coefficients including Half bit coefficients and low resolution coefficients/pixels. The transform step is efficiently implemented in integers as described in §3.

[0368] In step 604, the subband coefficients that are calculated in step 603 are variable length encoded and stored in the cache 121. If ~~maxBitPlane (tile)=0~~ maxBitPlane(tile)=0 we do not write any data. Else we loop on the coefficient groups $\{coef(2 \times i + x, 2 \times j + y)\}_{x,y=0,1}$. For each such group we first write the group's variable length ~~length (i,j)~~ length(i,j) using $\log_2(maxBitPlane(tile))$ bits. Then for each coefficient in the group we write ~~length (i,j)+1~~ length(i,j)+1 bits representing the coefficient's value. The least significant bit represents the coefficient's sign: if it is 1 then the variable length encoded coefficient is assumed to be negative. The HalfBit subband coefficients are written ~~in one bit~~ one bit per coefficient.

[0370] In step 503, unless losslessMode is true, the subband coefficients calculated in step 604 are quantized. This procedure is performed at this time ~~for the following reason: It~~ because it is required that the coefficients computed in the previous step [[will]] be stored in the cache 121. To avoid writing huge amounts of data to the cache, some compression is required. Thus, the quantization step serves as a preparation step for the ~~next~~ following variable length encoding step. It is important to point out that the quantization step has no effect on compression results. Namely, the quantization step is synchronized with the encoding algorithm such that the results of the encoding algorithm of quantized and non-quantized coefficients are identical.

[0371] A tile of an image component c at the resolution j is quantized using the given threshold ~~threshold (c,j)~~ threshold(c,j): for each ~~coefficients~~ coefficient x, the quantized value is $\lfloor x/threshold(c, j) \rfloor$. It is advantageous to choose the parameters ~~threshold (c,j)~~ threshold(c,j) to be dyadic such that the quantization can be implemented using integer shifts. The quantization procedure performed on a subband tile is as follows:

[0382] Step 503, [[is]] described in FIG. 7. ~~It~~ is only performed ~~whenever~~ when the data blocks associated with a low-resolution subband tile are not available in the server cache 121.

[0394] The present invention has been described in only a few embodiments, and with respect to only a few applications (e.g., commercial printing and medical imaging). Those of ordinary skill in the art will recognize that the teachings of the present invention may be used in a variety of other applications where images are to be transmitted over a communication media.